dhirajiitbhumech123 / **Projects** Public

Code Issues Pull requests Actions Projects 1 Wiki Security Insights Settings

master

Projects / Grid vs Random Search vs Bayesian Optimization.ipynb

dhirajiitbhumech123 Add files via upload History

1 contributor

755 lines (755 sloc) | 38 KB

Import the required modules

In [2]:
```
import os
import math
import numpy as np
from scipy.io import loadmat
import pandas as pd
import matplotlib.pyplot as plt
from functools import partial
from bayes_opt import BayesianOptimization

# Keras modules
from keras.preprocessing.image import ImageDataGenerator, array_to_i
mg, img_to_array, load_img
from keras.utils import to_categorical
from keras.models import Sequential
from keras import optimizers
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers import Activation, Dropout, Flatten, Dense
```

```
/home/poc/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36:
FutureWarning: Conversion of the second argument of issubdtype from
`float` to `np.floating` is deprecated. In future, it will be treate
d as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Pre-process the Dataset

In [3]:
```
# Load the datasets
sample_full_data = loadmat('Digit_Dataset_Full.mat')
label_train_data = pd.read_csv("Digit_Dataset_Full_Train_Labels.csv"
)
label_test_data = pd.read_csv("Digit_Dataset_Full_Test_Labels.csv")

# Get data from the datasets
X_train_orig = sample_full_data['Image'][0, 0][0]
X_test_orig = sample_full_data['Image'][0, 0][1]
```

```
Y_train_orig = label_train_data.values[:, 0]
Y_test_orig = label_test_data.values[:, 0]

# Print details of the orignal data
print("X_train_orig shape: " + str(X_train_orig.shape))
print("X_test_orig shape: " + str(X_test_orig.shape))
print("Y_train_orig shape: " + str(Y_train_orig.shape))
print("Y_test_orig shape: " + str(Y_test_orig.shape), "\n")

# Reshape the input data for keras
split_fraction = 0.9     # should be greater than 0.5
train_set_len = math.ceil((X_train_orig.shape[3] + X_test_orig.shape
[3]) * split_fraction)
test_set_len = X_train_orig.shape[3] + X_test_orig.shape[3] - train_
set_len
X_train = np.zeros((train_set_len, X_train_orig.shape[0], X_train_or
ig.shape[1], X_train_orig.shape[2]))
X_test = np.zeros((test_set_len, X_train_orig.shape[0], X_train_orig
.shape[1], X_train_orig.shape[2]))
Y_train = np.zeros((train_set_len, 1))
Y_test = np.zeros((test_set_len, 1))

# Split into train and test, by the given split fraction
for i in range(train_set_len + test_set_len):
    if i < train_set_len:
        if i < X_train_orig.shape[3]:
            X_train[i] = X_train_orig[:, :, :, i]
            Y_train[i] = Y_train_orig[i]
        else:
            X_train[i] = X_test_orig[:, :, :, i - X_train_orig.shape
[3]]
            Y_train[i] = Y_test_orig[i - X_train_orig.shape[3]]

    else:
        if i < X_train_orig.shape[3]:
            X_test[i - train_set_len] = X_train_orig[:, :, :, i]
            Y_test[i - train_set_len] = Y_train_orig[i]
        else:
            X_test[i - train_set_len] = X_test_orig[:, :, :, i - X_t
rain_orig.shape[3]]
            Y_test[i - train_set_len] = Y_test_orig[i - X_train_orig
.shape[3]]

# Convert the integer labels into one-hot
```

```
# Convert the integer labels into one hot
Y_train = to_categorical(Y_train, num_classes=10, dtype='float32')
Y_test = to_categorical(Y_test, num_classes=10, dtype='float32')

# Print details of the reshaped data
print("X_train shape: " + str(X_train.shape))
print("X_test shape: " + str(X_test.shape))
print("Y_train shape: " + str(Y_train.shape))
print("Y_test shape: " + str(Y_test.shape))

# Create an image generator class
imgGentrain = ImageDataGenerator()
imgGentest = ImageDataGenerator()

train_batch = imgGentrain.flow(
    x=X_train,
    y=Y_train,
    batch_size=32,
    shuffle=True,
    seed=1)
test_batch = imgGentest.flow(
    x=X_test,
    y=Y_test,
    batch_size=32,
    shuffle=True,
    seed=2)
```

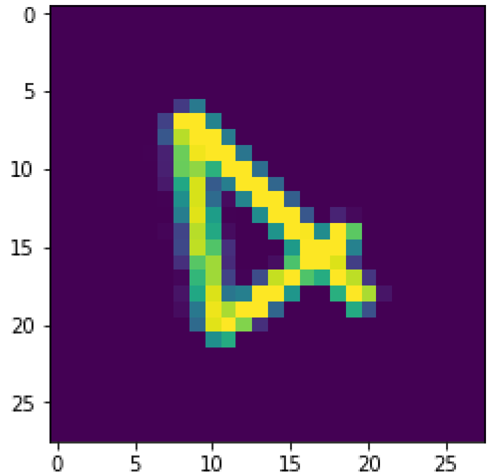
```
X_train_orig shape: (28, 28, 1, 9000)
X_test_orig shape: (28, 28, 1, 1000)
Y_train_orig shape: (9000,)
Y_test_orig shape: (1000,)

X_train shape: (9000, 28, 28, 1)
X_test shape: (1000, 28, 28, 1)
Y_train shape: (9000, 10)
Y_test shape: (1000, 10)
```

Example of an Image

In [4]:
```
index = 3600    # just some image for preview
plt.imshow(X_train[index, :, :, 0])
print("y = " + str(np.squeeze(Y_train[index, :])))
```

```
y = [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



1.1 - Create a Model (With Batch Normalization)

In [5]:
```
def create_model(lr=0.001, beta_1=0.9):
    try:
        del model
    except:
        pass

    # Create a model
    model = Sequential()

    # Add a convolutional layer
    model.add(Conv2D(filters=32, kernel_size=5, strides=(1, 1), padd
ing='valid', input_shape=(28, 28, 1)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding
='valid'))

    # 2. Add a convolution layer
    model.add(Conv2D(filters=16, kernel_size=3, strides=(1, 1), padd
ing='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))
```

```
    # Flatten the output
    model.add(Flatten())

    # Add a dense layer
    model.add(Dense(10))
    model.add(Activation('softmax'))
    optAdam = optimizers.Adam(lr=lr, beta_1=beta_1, beta_2=0.999, ep
silon=None, decay=0.0, amsgrad=False)
    model.compile(optimizer=optAdam,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Function to Evaluate the Current Model

In [6]:
```
def fit_with(lr=0.001, beta_1=0.9):
    # Create the model using a specified hyperparameters.
    model = create_model(lr=lr, beta_1=beta_1)

    # Train the model with the train dataset.
    model.fit_generator(
        generator=train_batch,
        steps_per_epoch=len(train_batch),
        epochs=3)

    # print the test accuracy
    score = model.evaluate(X_test, Y_test, verbose = 0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])

    # return the test accuracy
    return score[1]

fit_with_partial = partial(fit_with)
```

1.2 - Bayesian Optimization

In [7]:
```
# Bounded region of parameter space
```

```
pbounds = {'lr': (1e-4, 1e-2), 'beta_1': (0.8, 1)}

# Create the bayesian optimizer
optimizer = BayesianOptimization(
    f=fit_with_partial,
    pbounds=pbounds,
    verbose=2,  # verbose = 1 prints only when a maximum is observe
d, verbose = 0 is silent
    random_state=1,
)

# Maximize the accuracy
optimizer.maximize(init_points=10, n_iter=10,)

# Print the result
for i, res in enumerate(optimizer.res):
    print("Iteration {}: \n\t{}".format(i, res))

print(optimizer.max)
```

```
|   iter    |  target   |  beta_1   |    lr     |
-------------------------------------------------
WARNING:tensorflow:From /home/poc/anaconda3/lib/python3.6/site-packa
ges/tensorflow/python/framework/op_def_library.py:263: colocate_with
(from tensorflow.python.framework.ops) is deprecated and will be rem
oved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/poc/anaconda3/lib/python3.6/site-packa
ges/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflo
w.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/3
282/282 [==============================] - 21s 74ms/step - loss: 3.6
921 - acc: 0.6659
Epoch 2/3
282/282 [==============================] - 18s 62ms/step - loss: 3.2
720 - acc: 0.7848
Epoch 3/3
282/282 [==============================] - 17s 61ms/step - loss: 2.3
819 - acc: 0.8241
Test loss: 0.09564479201845824
```

```
Test accuracy: 0.971
|  1         |  0.971     |  0.8834    |  0.007231 |
Epoch 1/3
282/282 [==============================] - 18s 64ms/step - loss: 0.5
866 - acc: 0.8183
Epoch 2/3
282/282 [==============================] - 22s 78ms/step - loss: 0.0
882 - acc: 0.9709
Epoch 3/3
282/282 [==============================] - 24s 86ms/step - loss: 0.0
131 - acc: 0.9977
Test loss: 0.006796402305364609
Test accuracy: 0.998
|  2         |  0.998     |  0.8       |  0.003093 |
Epoch 1/3
282/282 [==============================] - 20s 69ms/step - loss: 0.6
574 - acc: 0.7941
Epoch 2/3
282/282 [==============================] - 25s 89ms/step - loss: 0.0
981 - acc: 0.9802
Epoch 3/3
282/282 [==============================] - 20s 70ms/step - loss: 0.0
341 - acc: 0.9960
Test loss: 0.030636158142238856
Test accuracy: 0.995
|  3         |  0.995     |  0.8294    |  0.001014 |
Epoch 1/3
282/282 [==============================] - 19s 69ms/step - loss: 0.6
193 - acc: 0.8108
Epoch 2/3
282/282 [==============================] - 21s 73ms/step - loss: 0.0
577 - acc: 0.9852
Epoch 3/3
282/282 [==============================] - 19s 68ms/step - loss: 0.0
401 - acc: 0.9888
Test loss: 0.009827798534883187
Test accuracy: 0.997
|  4         |  0.997     |  0.8373    |  0.003521 |
Epoch 1/3
282/282 [==============================] - 18s 65ms/step - loss: 0.6
721 - acc: 0.8057
Epoch 2/3
282/282 [==============================] - 16s 55ms/step - loss: 0.0
679 - acc: 0.9792
```

```
Epoch 3/3
282/282 [==============================] - 15s 52ms/step - loss: 0.0
723 - acc: 0.9778
Test loss: 0.12063172279548598
Test accuracy: 0.963
|  5         |  0.963    |  0.8794   |  0.005434 |
Epoch 1/3
282/282 [==============================] - 16s 57ms/step - loss: 2.2
387 - acc: 0.7450
Epoch 2/3
282/282 [==============================] - 15s 52ms/step - loss: 0.0
764 - acc: 0.9764
Epoch 3/3
282/282 [==============================] - 15s 52ms/step - loss: 0.0
360 - acc: 0.9906
Test loss: 0.03859443750977516
Test accuracy: 0.994
|  6         |  0.994    |  0.8838   |  0.006884 |
Epoch 1/3
282/282 [==============================] - 16s 56ms/step - loss: 1.7
068 - acc: 0.7617
Epoch 2/3
282/282 [==============================] - 15s 52ms/step - loss: 0.0
918 - acc: 0.9726
Epoch 3/3
282/282 [==============================] - 15s 53ms/step - loss: 0.0
206 - acc: 0.9945
Test loss: 0.038596218196558764
Test accuracy: 0.991
|  7         |  0.991    |  0.8409   |  0.008793 |
Epoch 1/3
282/282 [==============================] - 22s 76ms/step - loss: 0.7
717 - acc: 0.8208
Epoch 2/3
282/282 [==============================] - 19s 67ms/step - loss: 0.0
764 - acc: 0.9751
Epoch 3/3
282/282 [==============================] - 19s 67ms/step - loss: 0.0
402 - acc: 0.9876
Test loss: 0.0062416974066291
Test accuracy: 0.998
|  8         |  0.998    |  0.8055   |  0.006738 |
Epoch 1/3
```

```
282/282 [==============================] - 20s 71ms/step - loss: 0.6
634 - acc: 0.8142
Epoch 2/3
282/282 [==============================] - 19s 67ms/step - loss: 0.0
951 - acc: 0.9709
Epoch 3/3
282/282 [==============================] - 19s 67ms/step - loss: 0.0
266 - acc: 0.9919
Test loss: 0.02194678747165017
Test accuracy: 0.994
|  9        |  0.994    |  0.8835   |  0.005631 |
Epoch 1/3
282/282 [==============================] - 20s 72ms/step - loss: 0.4
977 - acc: 0.8445
Epoch 2/3
282/282 [==============================] - 19s 69ms/step - loss: 0.0
648 - acc: 0.9827
Epoch 3/3
282/282 [==============================] - 20s 69ms/step - loss: 0.0
497 - acc: 0.9857
Test loss: 0.11759197735682392
Test accuracy: 0.969
|  10       |  0.969    |  0.8281   |  0.002061 |
Epoch 1/3
282/282 [==============================] - 22s 76ms/step - loss: 2.4
398 - acc: 0.6154
Epoch 2/3
282/282 [==============================] - 15s 53ms/step - loss: 0.1
337 - acc: 0.9574
Epoch 3/3
282/282 [==============================] - 15s 53ms/step - loss: 0.0
635 - acc: 0.9792
Test loss: 0.03985049867187627
Test accuracy: 0.987
|  11       |  0.987    |  0.9238   |  0.01     |
Epoch 1/3
282/282 [==============================] - 16s 58ms/step - loss: nan
- acc: 0.1018
Epoch 2/3
282/282 [==============================] - 15s 53ms/step - loss: nan
- acc: 0.0987
Epoch 3/3
282/282 [==============================] - 15s 53ms/step - loss: nan
- acc: 0.0985
```

```
Test loss: nan
Test accuracy: 0.1
|  12       |  0.1      |  1.0      |  0.0001   |
Epoch 1/3
282/282 [==============================] - 17s 61ms/step - loss: 1.6
894 - acc: 0.4381
Epoch 2/3
282/282 [==============================] - 15s 54ms/step - loss: 0.8
803 - acc: 0.7475
Epoch 3/3
282/282 [==============================] - 15s 54ms/step - loss: 0.5
482 - acc: 0.8516
Test loss: 0.4763501634597778
Test accuracy: 0.886
|  13       |  0.886    |  0.9575   |  0.0001   |
Epoch 1/3
282/282 [==============================] - 17s 62ms/step - loss: 1.6
713 - acc: 0.4390
Epoch 2/3
282/282 [==============================] - 15s 54ms/step - loss: 0.8
608 - acc: 0.7570
Epoch 3/3
282/282 [==============================] - 15s 55ms/step - loss: 0.5
507 - acc: 0.8598
Test loss: 0.45985244297981265
Test accuracy: 0.882
|  14       |  0.882    |  0.9061   |  0.0001   |
Epoch 1/3
282/282 [==============================] - 17s 61ms/step - loss: 1.6
459 - acc: 0.4539
Epoch 2/3
282/282 [==============================] - 20s 71ms/step - loss: 0.8
536 - acc: 0.7550
Epoch 3/3
282/282 [==============================] - 21s 75ms/step - loss: 0.5
326 - acc: 0.8710
Test loss: 0.46304218673706055
Test accuracy: 0.889
|  15       |  0.889    |  0.8584   |  0.0001   |
Epoch 1/3
282/282 [==============================] - 22s 76ms/step - loss: 1.8
277 - acc: 0.6468
Epoch 2/3
```

```
282/282 [==============================] - 20s 71ms/step - loss: 0.1
705 - acc: 0.9496
Epoch 3/3
282/282 [==============================] - 19s 68ms/step - loss: 0.0
803 - acc: 0.9768
Test loss: 0.07756807271763683
Test accuracy: 0.964
|  16       |  0.964    |  0.9746   |  0.01     |
Epoch 1/3
282/282 [==============================] - 21s 73ms/step - loss: 4.3
146 - acc: 0.4450
Epoch 2/3
282/282 [==============================] - 20s 72ms/step - loss: 0.3
303 - acc: 0.8924
Epoch 3/3
282/282 [==============================] - 23s 83ms/step - loss: 0.1
084 - acc: 0.9678
Test loss: 0.14060125095583498
Test accuracy: 0.956
|  17       |  0.956    |  0.9445   |  0.01     |
Epoch 1/3
282/282 [==============================] - 24s 85ms/step - loss: 2.3
155 - acc: 0.7216
Epoch 2/3
282/282 [==============================] - 20s 70ms/step - loss: 1.7
141 - acc: 0.8757
Epoch 3/3
282/282 [==============================] - 20s 71ms/step - loss: 1.6
485 - acc: 0.8889
Test loss: 1.622260201461293
Test accuracy: 0.899
|  18       |  0.899    |  0.9013   |  0.01     |
Epoch 1/3
282/282 [==============================] - 20s 72ms/step - loss: 1.7
040 - acc: 0.4383
Epoch 2/3
282/282 [==============================] - 19s 66ms/step - loss: 0.8
624 - acc: 0.7502
Epoch 3/3
282/282 [==============================] - 18s 64ms/step - loss: 0.5
462 - acc: 0.8610
Test loss: 0.4608350868225098
Test accuracy: 0.878
|  19       |  0.878    |  0.9335   |  0.0001   |
```

```
Epoch 1/3
282/282 [==============================] - 20s 72ms/step - loss: 1.5
992 - acc: 0.4848
Epoch 2/3
282/282 [==============================] - 18s 65ms/step - loss: 0.8
468 - acc: 0.7564
Epoch 3/3
282/282 [==============================] - 18s 63ms/step - loss: 0.5
337 - acc: 0.8681
Test loss: 0.4637343616485596
Test accuracy: 0.88
|  20       |  0.88     |  0.9763   |  0.0001   |
=================================================
Iteration 0:
        {'target': 0.971, 'params': {'beta_1': 0.8834044009405149,
 'lr': 0.007231212485077366}}
Iteration 1:
        {'target': 0.998, 'params': {'beta_1': 0.800022874963469, 'l
r': 0.003093092469055214}}
Iteration 2:
        {'target': 0.995, 'params': {'beta_1': 0.8293511781634226,
 'lr': 0.0010141520882110983}}
Iteration 3:
        {'target': 0.997, 'params': {'beta_1': 0.8372520422755342,
 'lr': 0.003521051197726173}}
Iteration 4:
        {'target': 0.963, 'params': {'beta_1': 0.879353494846134, 'l
r': 0.005434285666633234}}
Iteration 5:
        {'target': 0.994, 'params': {'beta_1': 0.883838902880659, 'l
r': 0.00688367305392792}}
Iteration 6:
        {'target': 0.991, 'params': {'beta_1': 0.8408904499463035,
 'lr': 0.00879336262027036}}
Iteration 7:
        {'target': 0.998, 'params': {'beta_1': 0.8054775186395853,
 'lr': 0.006737628350766182 4}}
Iteration 8:
        {'target': 0.994, 'params': {'beta_1': 0.8834609604734254,
 'lr': 0.005631029301612942}}
Iteration 9:
        {'target': 0.969, 'params': {'beta_1': 0.8280773877190468,
 'lr': 0.0020612047419403}}
```

```
Iteration 10:
        {'target': 0.987, 'params': {'beta_1': 0.9238248606315559,
 'lr': 0.01}}
Iteration 11:
        {'target': 0.1, 'params': {'beta_1': 1.0, 'lr': 0.0001}}
Iteration 12:
        {'target': 0.886, 'params': {'beta_1': 0.9574847719448162,
 'lr': 0.0001}}
Iteration 13:
        {'target': 0.882, 'params': {'beta_1': 0.9060606688478187,
 'lr': 0.0001}}
Iteration 14:
        {'target': 0.889, 'params': {'beta_1': 0.8583786436774691,
 'lr': 0.0001}}
Iteration 15:
        {'target': 0.964, 'params': {'beta_1': 0.974588072127511, 'l
r': 0.01}}
Iteration 16:
        {'target': 0.956, 'params': {'beta_1': 0.9444577517990637,
 'lr': 0.01}}
Iteration 17:
        {'target': 0.899, 'params': {'beta_1': 0.901313725393386, 'l
r': 0.01}}
Iteration 18:
        {'target': 0.878, 'params': {'beta_1': 0.9334773444036866,
 'lr': 0.0001}}
Iteration 19:
        {'target': 0.88, 'params': {'beta_1': 0.9763409771195536, 'l
r': 0.0001}}
{'target': 0.998, 'params': {'beta_1': 0.800022874963469, 'lr': 0.00
3093092469055214}}
```

1.3 - Perform Grid Search

In [8]:
```
lr_list = np.array([10**-5, 10**-4, 10**-3])
mom_list = np.array([0.8, 0.9, 1.0])

g_grid = np.meshgrid(lr_list, mom_list)
g_grid_points = np.append(g_grid[0].reshape(-1,1), g_grid[1].reshape
(-1,1), axis=1)

g_result_list = []
```

```
for i in g_grid_points:
    model = create_model(lr=i[0], beta_1=i[1])
    model.fit_generator(
        generator=train_batch,
        steps_per_epoch=len(train_batch),
        epochs=5)
    score = model.evaluate(X_test, Y_test, verbose = 0)
    g_result_list.append([i[0], i[1], score[0], score[1]])

for i in g_result_list:
    print("For learning rate = {0} and momentum = {1}: loss = {2}, a
ccuracy = {3}".format(i[0], i[1], i[2], i[3]))
```

```
Epoch 1/5
282/282 [==============================] - 22s 77ms/step - loss: 2.4
378 - acc: 0.1541
Epoch 2/5
282/282 [==============================] - 18s 63ms/step - loss: 2.1
100 - acc: 0.2493
Epoch 3/5
282/282 [==============================] - 19s 68ms/step - loss: 1.8
855 - acc: 0.3478
Epoch 4/5
282/282 [==============================] - 19s 69ms/step - loss: 1.6
913 - acc: 0.4373
Epoch 5/5
282/282 [==============================] - 19s 69ms/step - loss: 1.5
242 - acc: 0.5178
Epoch 1/5
282/282 [==============================] - 18s 64ms/step - loss: 1.6
936 - acc: 0.4367
Epoch 2/5
282/282 [==============================] - 15s 54ms/step - loss: 0.8
455 - acc: 0.7609
Epoch 3/5
```